LAB 01

COMPUTER ORGANIZATION AND ASSEMBLY LANG(COAL)

Sarim Latif Khan	20K-1644	3E
STUDENT NAME	ROLL NO	SEC

SIGNATURE & DATE

MARKS AWARDED: ______________

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES (NUCES), KARACHI

Prepared by: Amin Sadiq

Version: 1.0

Date:

EL 2003 COMP ORG & ASSEMBLY LANGUAGE LAB

LABORATORY MANUAL Fall 2021Instructor: Aamir Ali, Aashir Mehboob, Amin Sadiq, Qurat-ul-Ain, M Kariz, Rabia

Lab Session 01 CONFIGURATION OF VS 2019

Lab Session 01: CONFIGURATION OF VS 2019

Objectives:

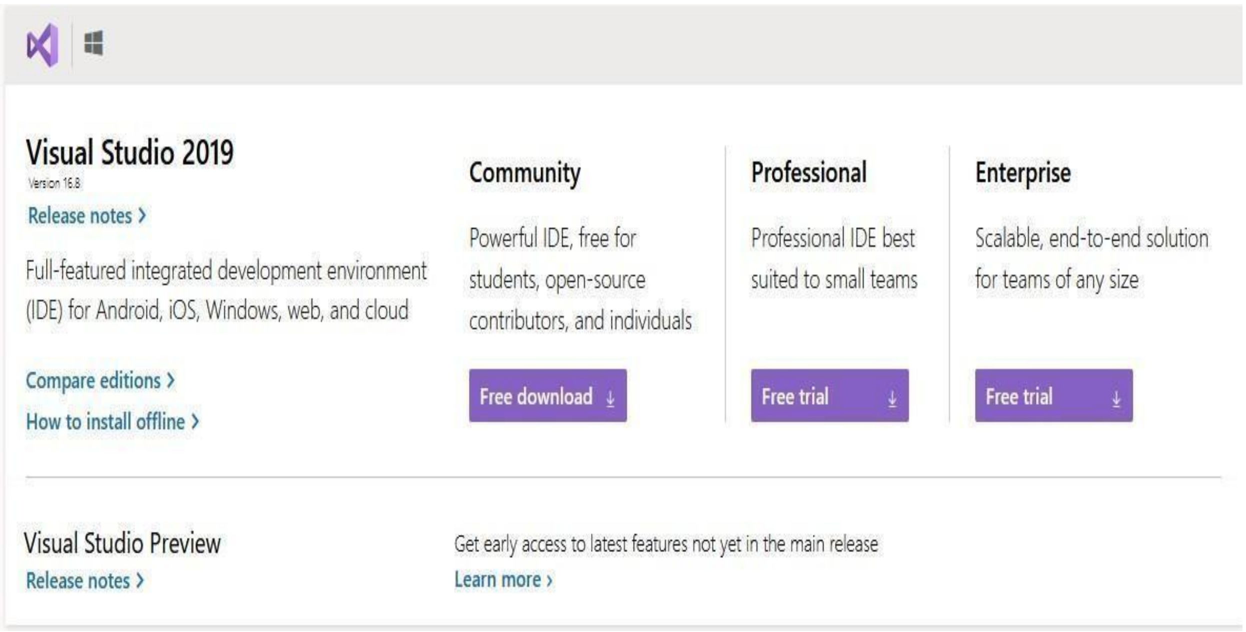
- Introduction to Visual Studio 2019
- An introduction to Assembly Language
- Understanding the Visual Studio 2019
- Configuring Visual Studio 2019 to activate MASM assembler
- Running a test program

SECTION 1: INTRODUCTION TO VISUAL STUDIO 2019

Visual Studio (for all its versions) is an integrated development environment (IDE) from Microsoft. It is a leading tool to develop computer programs, as well as web sites, web applications and web services. For this course, we will use Visual Studio version 2019 to develop programs in Assembly Language. We could, however, use a stand-alone assembler like NASM or MASM to code in Assembly Language.

INSTALLATION PROCESS

Go to this link https://visualstudio.microsoft.com/downloads/ and select VS 2019 Download for community version



Visual Studio 2019
Version 16.8
Release notes ›

Full-featured integrated development environment (IDE) for Android, iOS, Windows, web, and cloud

Compare editions ›
How to install offline ›

Community
Powerful IDE, free for students, open-source contributors, and individuals
Free download ↓

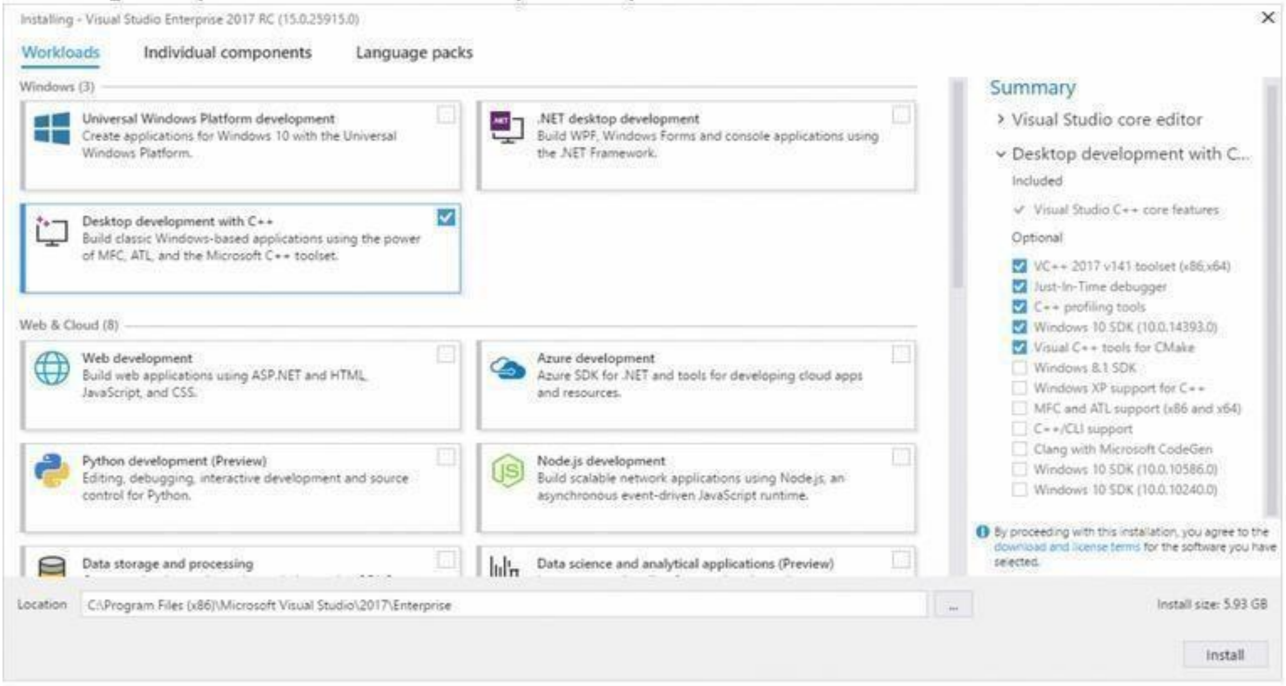
Professional
Professional IDE best suited to small teams
Free trial ↓

Enterprise
Scalable, end-to-end solution for teams of any size
Free trial ↓

Visual Studio Preview
Release notes ›

Get early access to latest features not yet in the main release
Learn more ›

Run that downloaded setup on your system and when it's complete, you have to download and install **Desktop Development with C++**. When it's done you are ready to go.



Installing - Visual Studio Enterprise 2017 RC (15.0.25915.0)

Workloads | Individual components | Language packs

Windows (3)

- Universal Windows Platform development — Create applications for Windows 10 with the Universal Windows Platform. ☐
- .NET desktop development — Build WPF, Windows Forms and console applications using the .NET Framework. ☐
- Desktop development with C++ — Build classic Windows-based applications using the power of MFC, ATL, and the Microsoft C++ toolset. ☑

Web & Cloud (8)

- Web development — Build web applications using ASP.NET and HTML, JavaScript, and CSS. ☐
- Azure development — Azure SDK for .NET and tools for developing cloud apps and resources. ☐
- Python development (Preview) — Editing, debugging, interactive development and source control for Python. ☐
- Node.js development — Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime. ☐
- Data storage and processing ☐
- Data science and analytical applications (Preview) ☐

Summary
› Visual Studio core editor
⌄ Desktop development with C...
Included
✓ Visual Studio C++ core features
Optional
☑ VC++ 2017 v141 toolset (x86,x64)
☑ Just-In-Time debugger
☑ C++ profiling tools
☑ Windows 10 SDK (10.0.14393.0)
☑ Visual C++ tools for CMake
☐ Windows 8.1 SDK
☐ Windows XP support for C++
☐ MFC and ATL support (x86 and x64)
☐ C++/CLI support
☐ Clang with Microsoft CodeGen
☐ Windows 10 SDK (10.0.10586.0)
☐ Windows 10 SDK (10.0.10240.0)

By proceeding with this installation, you agree to the download and license terms for the software you have selected.

Location: C:\Program Files (x86)\Microsoft Visual Studio\2017\Enterprise

Install size: 5.93 GB

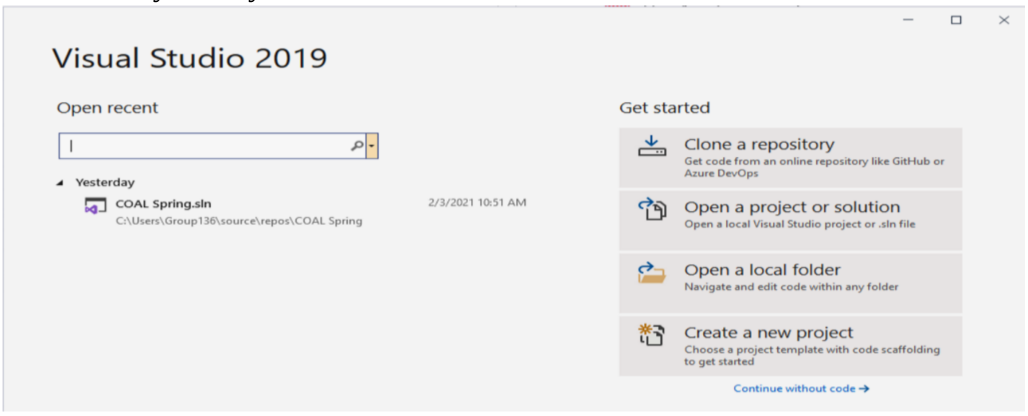
Install

CONFIGURATION VS2019 FOR ASSEMBLY LANGUAGE

Click here to download Irvine library from this link:www.asmirvine.com/gettingStartedVS2019/Irvine.zip

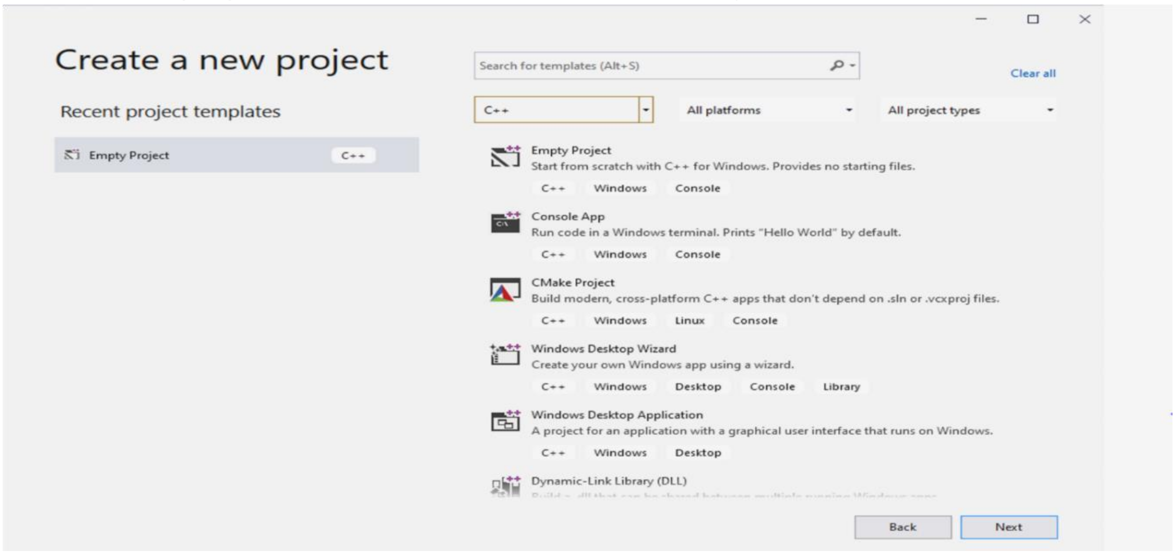
Once you have downloaded the required Irvine library, install it in your computer and verify that a folder named Irvine has been created in your C:\ drive. Now, follow these steps to configure Visual Studio 2019:

1. Start Microsoft Visual Studio 2019. If you are running it for the first time then this would be the screen you may see

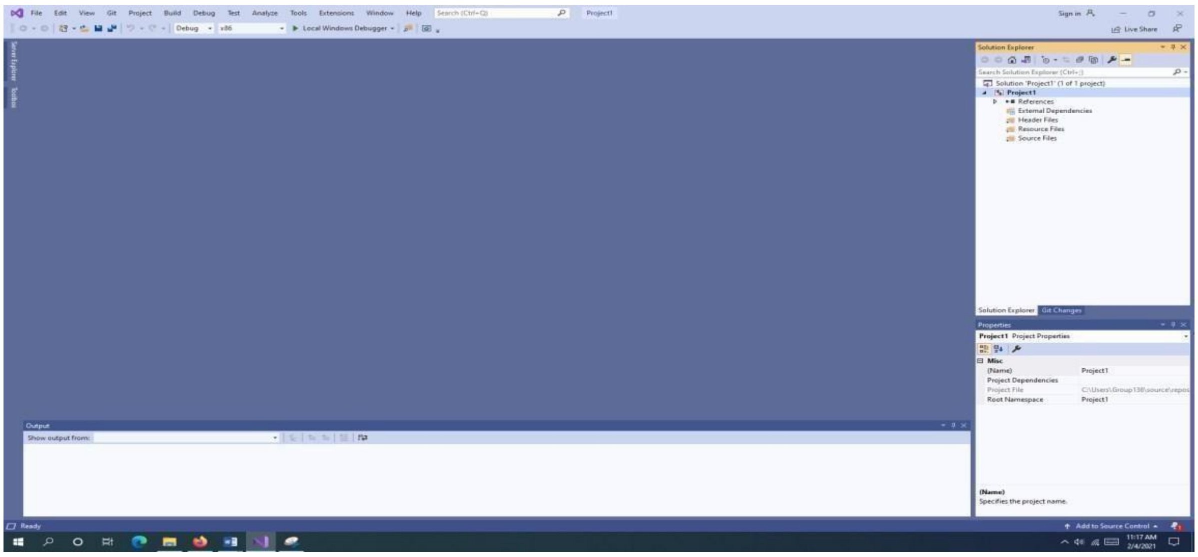

Visual Studio 2019
Open recent
Yesterday
COAL Spring.sln
C:\Users\Group136\source\repos\COAL Spring
2/3/2021 10:51 AM
Get started
Clone a repository
Get code from an online repository like GitHub or Azure DevOps
Open a project or solution
Open a local Visual Studio project or .sln file
Open a local folder
Navigate and edit code within any folder
Create a new project
Choose a project template with code scaffolding to get started
Continue without code →

Select Create a new project.

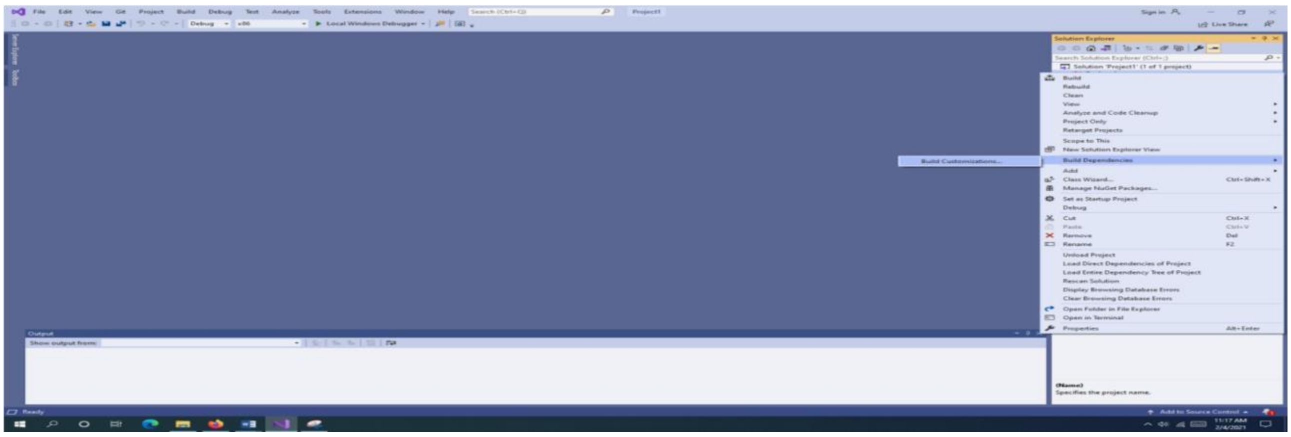
2. From languages select **C++,** and then create an empty project


Create a new project
Recent project templates
Empty Project C++
Search for templates (Alt+S)
Clear all
C++
All platforms
All project types
Empty Project
Start from scratch with C++ for Windows. Provides no starting files.
C++ Windows Console
Console App
Run code in a Windows terminal. Prints "Hello World" by default.
C++ Windows Console
CMake Project
Build modern, cross-platform C++ apps that don't depend on .sln or .vcxproj files.
C++ Windows Linux Console
Windows Desktop Wizard
Create your own Windows app using a wizard.
C++ Windows Desktop Console Library
Windows Desktop Application
A project for an application with a graphical user interface that runs on Windows.
C++ Windows Desktop
Dynamic-Link Library (DLL)
Back
Next

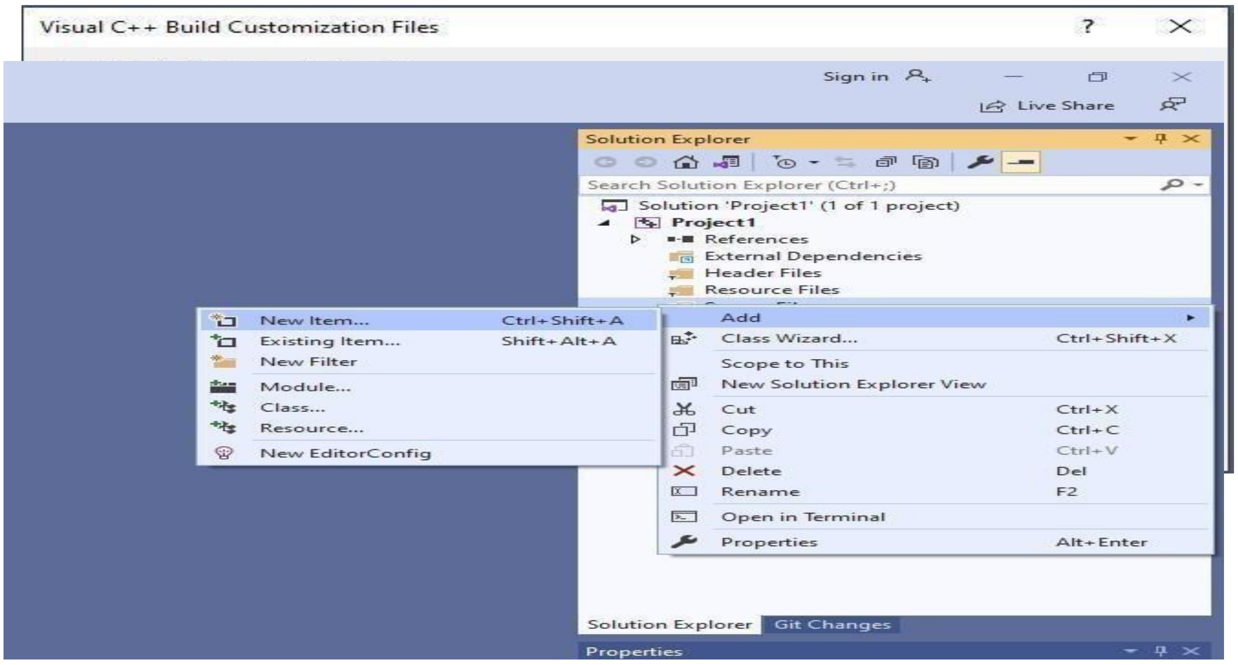
3. Once your new project is created, press **Ctrl+Alt+L** to open **Solution Explorer**. In the solution explorer window, you would see your project's file hierarchy.



Now right click on your project. Go to **Build Dependencies** and then select **Build Customization**

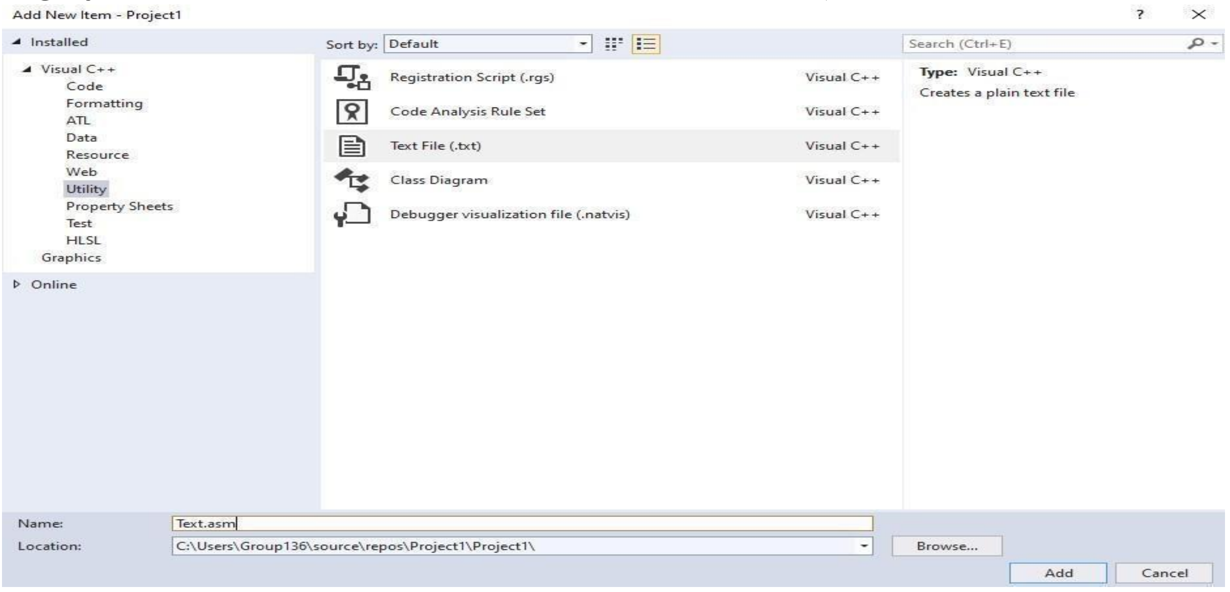


Tick the **masm** checkbox & select **OK**.



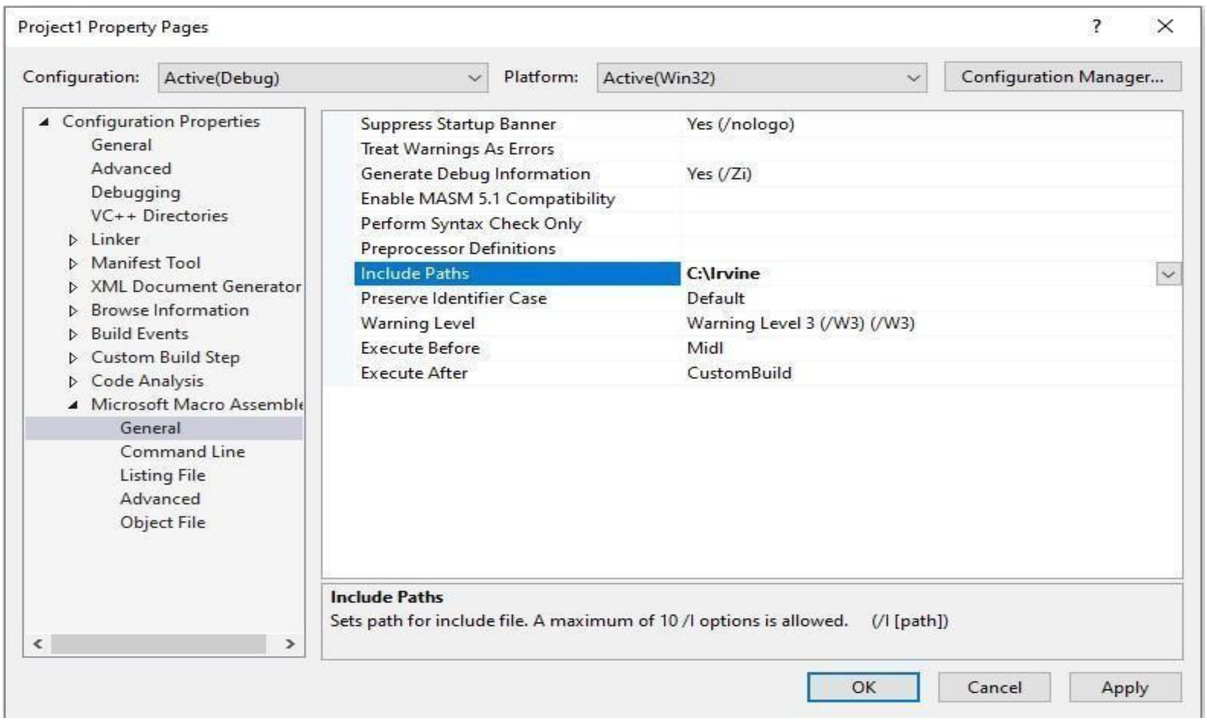
4. Right-click on **Source Files** in solution explorer & select **Add** > **New Item**.

Now go to **Utility** > **Text File** to add a new file, but we do not want to add .txt file, instead we want to add a .asm file. So, rename your new text file as Test.asm (we can choose any other name e.g. xyz.asm but for this tutorial we will use the name Test.asm).



Name: Text.asm
Location: C:\Users\Group136\source\repos\Project1\Project1\

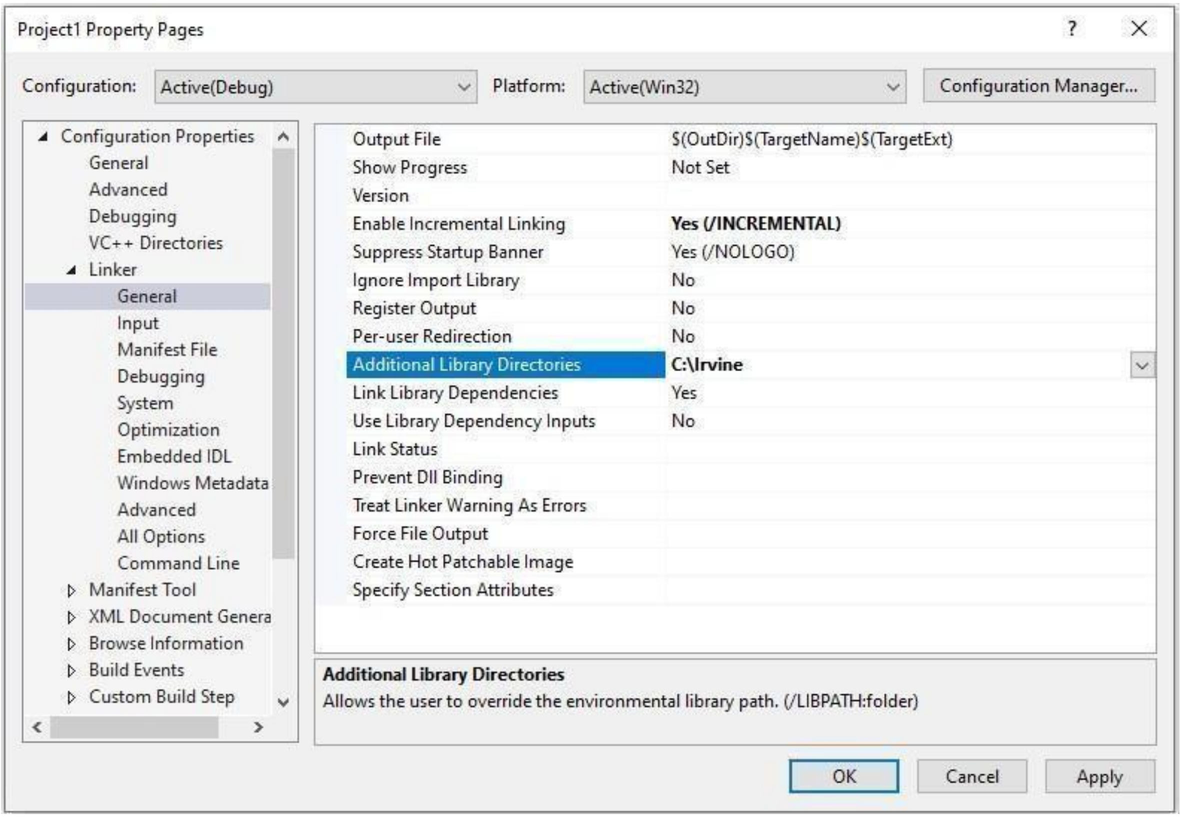
5. Now right-click your project again and click **Properties**. Now click the tiny arrow marker besides **Configuration Properties** to expand it. Now click **Microsoft Macro Assembler** and expand it.



Suppress Startup Banner	Yes (/nologo)
Treat Warnings As Errors	
Generate Debug Information	Yes (/Zi)
Enable MASM 5.1 Compatibility	
Perform Syntax Check Only	
Preprocessor Definitions	
Include Paths	C:\Irvine
Preserve Identifier Case	Default
Warning Level	Warning Level 3 (/W3) (/W3)
Execute Before	Midl
Execute After	CustomBuild

Include Paths
Sets path for include file. A maximum of 10 /I options is allowed. (/I [path])

6. Now click **General** entry under Microsoft Macro Assembler and then set the value of **Include Paths** as **C:\Irvine**. The menu should now like this.
7. Click **Linker** tab to expand it. Select **General** and set the value of **Additional Library Directories** to **C:\Irvine**



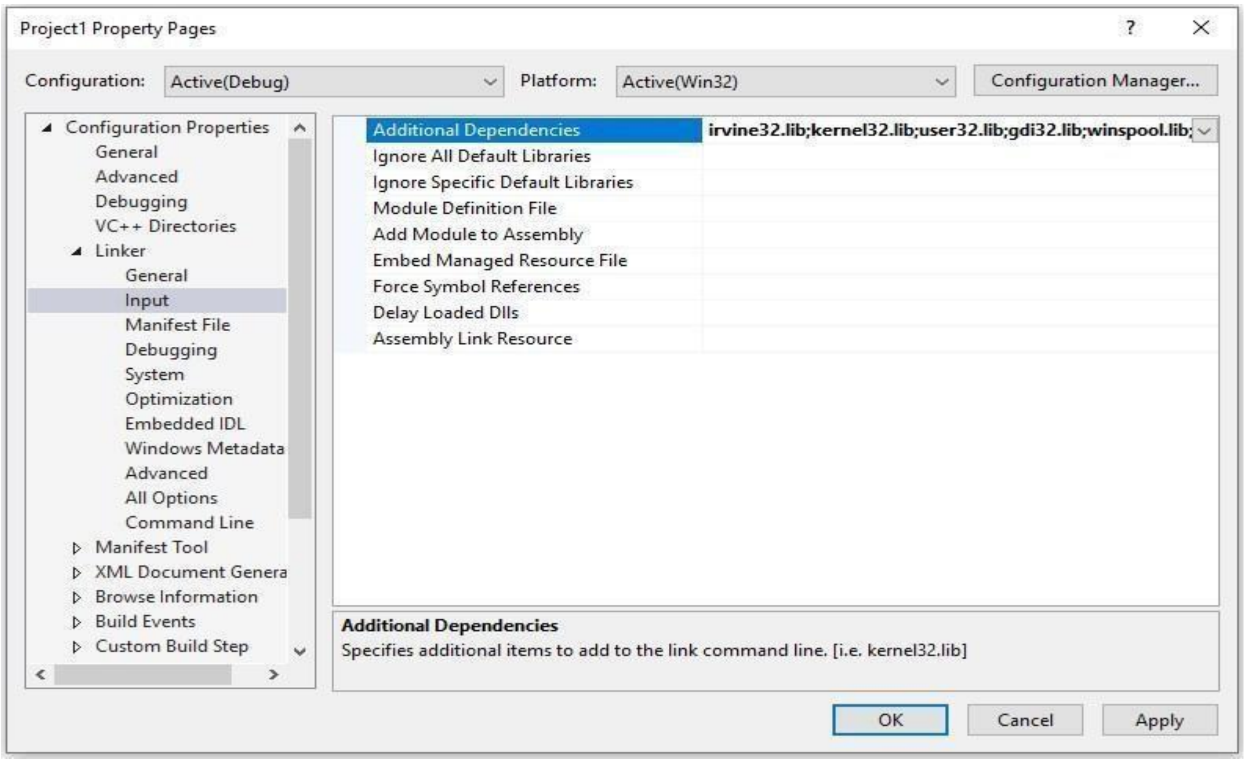
Output File	$(OutDir)$(TargetName)$(TargetExt)
Show Progress	Not Set
Version	
Enable Incremental Linking	**Yes (/INCREMENTAL)**
Suppress Startup Banner	Yes (/NOLOGO)
Ignore Import Library	No
Register Output	No
Per-user Redirection	No
Additional Library Directories	**C:\Irvine**
Link Library Dependencies	Yes
Use Library Dependency Inputs	No
Link Status	
Prevent Dll Binding	
Treat Linker Warning As Errors	
Force File Output	
Create Hot Patchable Image	
Specify Section Attributes	

Additional Library Directories
Allows the user to override the environmental library path. (/LIBPATH:folder)

Our Visual Studio 2019 configuration for Assembly Language is complete. We can now write a sample program and run it to test our project. Open Test.asm from the solution explorer by doubleclicking it. The Test.asm file will contain all the code that we write in our program. Go on and copy the following code onto your Test.asm file.

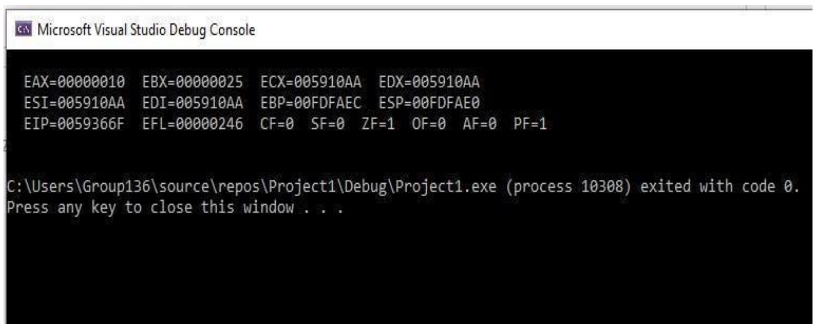
```
TITLE My First Program (Test.asm)
INCLUDE Irvine32.inc
```

8. Click **Input**, select **Additional Dependencies**. You will see a list of different .lib file names written there, do not alter any of those. Write **irvine32.lib;** at the start of the list like this.



Additional Dependencies: irvine32.lib;kernel32.lib;user32.lib;gdi32.lib;winspool.lib;

Additional Dependencies
Specifies additional items to add to the link command line. [i.e. kernel32.lib]



```
  EAX=00000010  EBX=00000025  ECX=005910AA  EDX=005910AA
  ESI=005910AA  EDI=005910AA  EBP=00FDFAEC  ESP=00FDFAE0
  EIP=0059366F  EFL=00000246  CF=0  SF=0  ZF=1  OF=0  AF=0  PF=1

C:\Users\Group136\source\repos\Project1\Debug\Project1.exe (process 10308) exited with code 0.
Press any key to close this window . . .
```

.code main

```
PROC mov eax,
10h mov
ebx, 25h call
DumpRegs exit
main ENDP
END main
```

Press **Ctrl+F5** to see the output in console window.
As we can see in the output window, the program has affected two registers eax & ebx. Let us dissect our code line by line to see what it does.

The first line TITLE MyFirstProgram (Test.asm) gives an optional title to our program. The second line INCLUDE irvine32.inc adds a reference to the include file that links your program to the Irvine library. The third line .code defines the beginning of the code segment (to be covered in detail later). The code segment is the segment of memory where all your code resides. In the fourth line, a main procedure is defined. The fifth and sixth lines show a mnemonic mov (to be covered in detail later) that 'moves' values 10h and 25h to eax and ebx, respectively. The radix h defines a hexadecimal constant.

The lines seven and eight calls the procedure DumpRegs that outputs the current values of the registers followed by a call to windows procedure named exit that halts the program. The lines nine and ten mark the end of the main procedure.

Section 2: Debugging our program

We have seen how to configure Visual Studio 2019 for Assembly Language and tested it with a sample program. The output of our sample program was displayed using a console window but it is usually more desirable to watch the step by step execution of our program with each line of code using breakpoints.

Let us briefly define the keywords relevant to debugging in Visual Studio and then we will cover an example for understanding.

Debugger

The (Visual Studio) debugger helps us observe the run-time behavior of our program and find problems. With the debugger, we can break execution of our program to examine our code, examine and edit variables, view registers, see the instructions created from our source code, and view the memory space used by our application.

Breakpoint

A breakpoint is a signal that tells the debugger to temporarily suspend execution of your program at a certain point. When execution is suspended at a breakpoint, your program is said to be in break mode.

Code Stepping

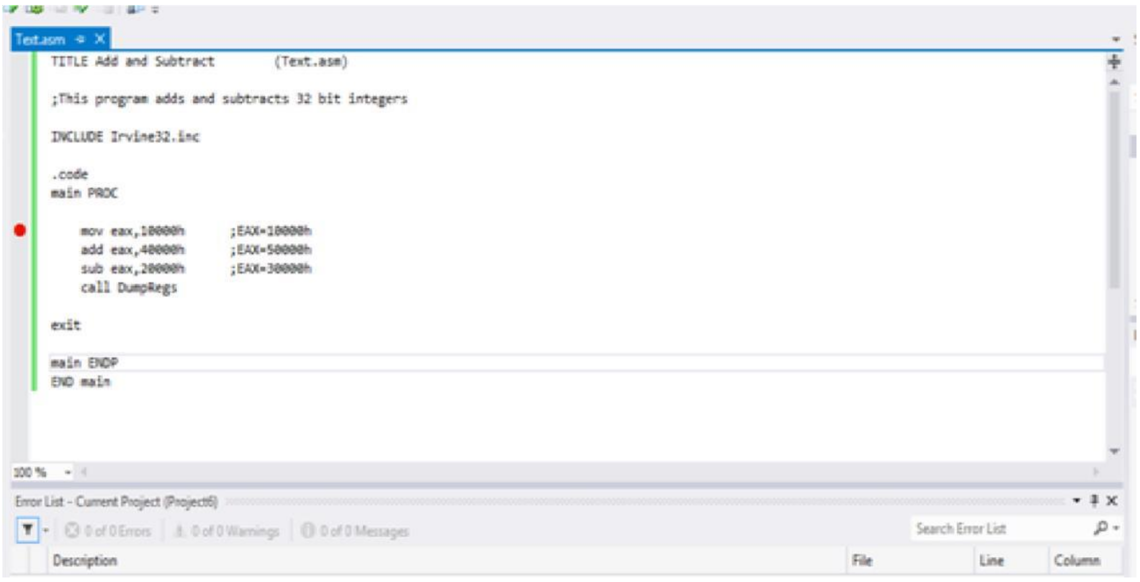
One of the most common debugging procedures is stepping: executing code one line at a time. The Debug menu provides three commands for stepping through code:

- Step Into (By pressing F11)
- Step Over (By pressing F10) ● Step Out (Shift+F11)

Single Stepping

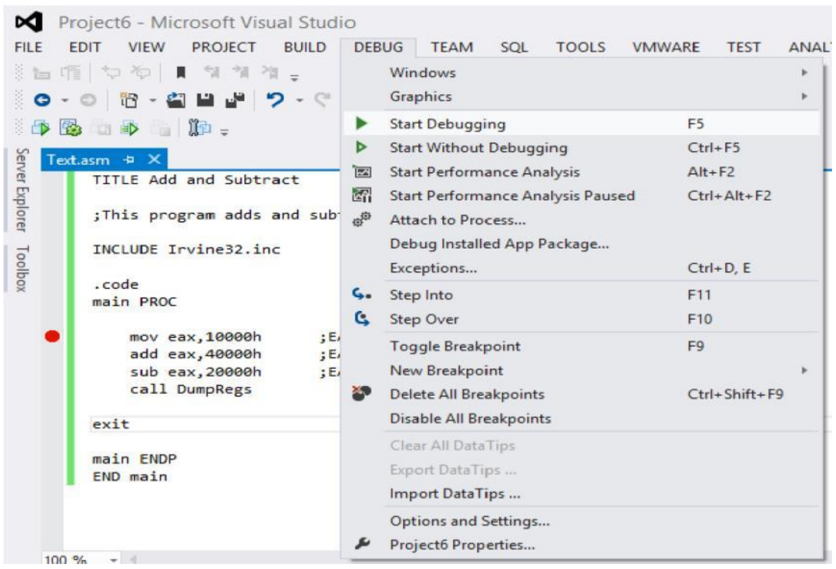
To see the values of internal registers and memory variables during execution, let us use an example. Copy the following code onto your Test.asm file.k

1. Right-click on line 6 to insert a breakpoint.



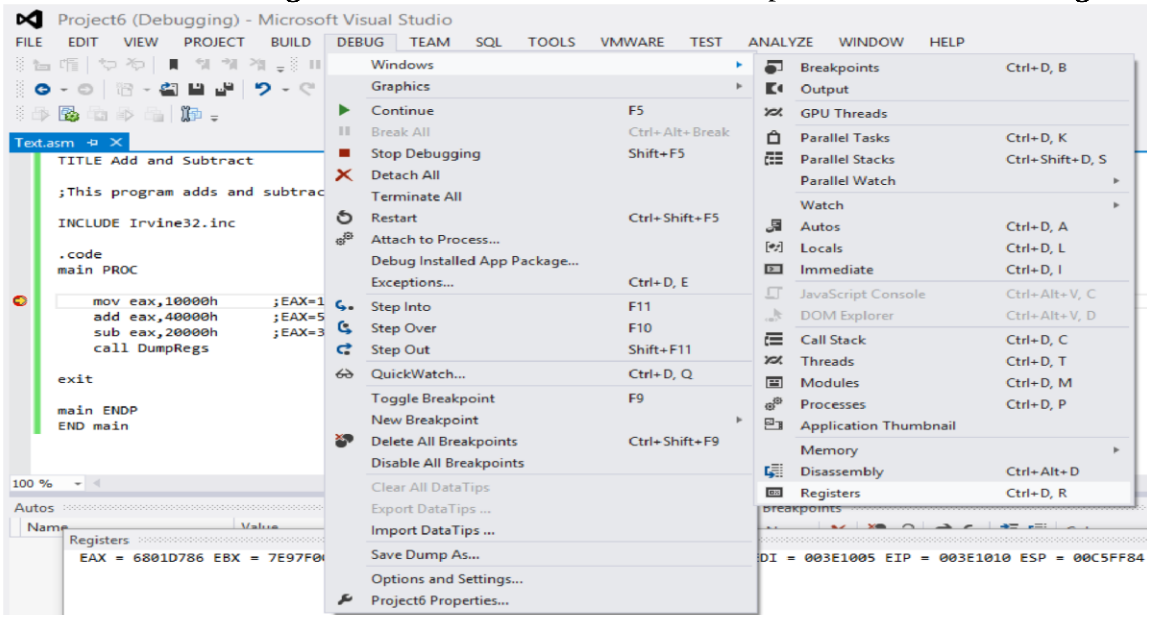
```
TITLE Add and Subtract        (Text.asm)

;This program adds and subtracts 32 bit integers

INCLUDE Irvine32.inc

.code
main PROC

    mov eax,10000h      ;EAX=10000h
    add eax,40000h      ;EAX=50000h
    sub eax,20000h      ;EAX=30000h
    call DumpRegs

exit

main ENDP
END main
```

2. Click on **Debug** tab from the toolbar, select **Start Debugging** OR press **F10** to start stepping over the code.



```
TITLE Add and Subtract

;This program adds and sub

INCLUDE Irvine32.inc

.code
main PROC

    mov eax,10000h      ;E
    add eax,40000h      ;E
    sub eax,20000h      ;E
    call DumpRegs

exit

main ENDP
END main
```

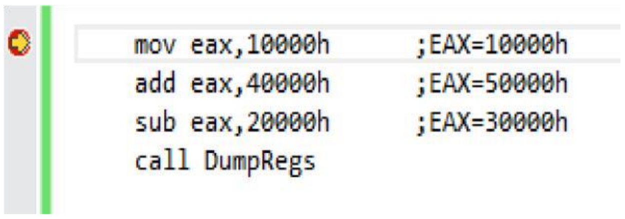
3. Click on **Debug** tab than select Windows after that open menu and select **Registers** option.



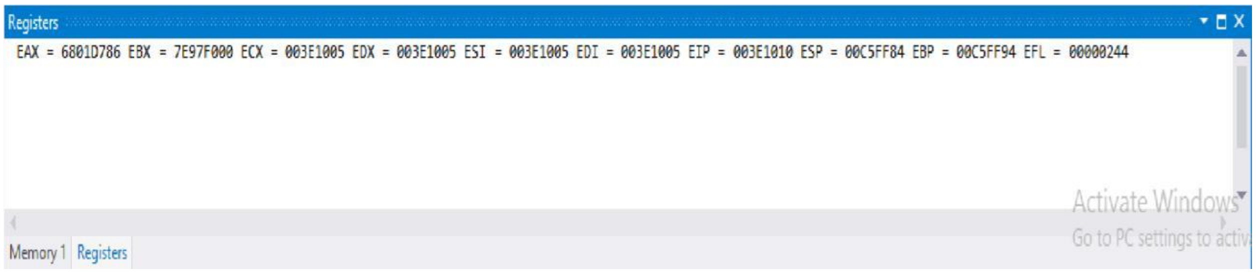
```
TITLE Add and Subtract

;This program adds and subtrac

INCLUDE Irvine32.inc

.code
main PROC

    mov eax,10000h          ;EAX=1
    add eax,40000h          ;EAX=5
    sub eax,20000h          ;EAX=3
    call DumpRegs

exit

main ENDP
END main
```

```
EAX = 6801D786 EBX = 7E97F0
DI = 003E1005 EIP = 003E1010 ESP = 00C5FF84
```

4. Breakpoint set on 1st instruction

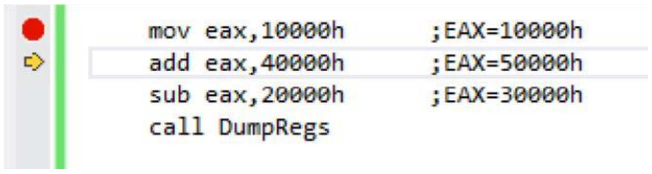


```
mov eax,10000h        ;EAX=10000h
add eax,40000h        ;EAX=50000h
sub eax,20000h        ;EAX=30000h
call DumpRegs
```

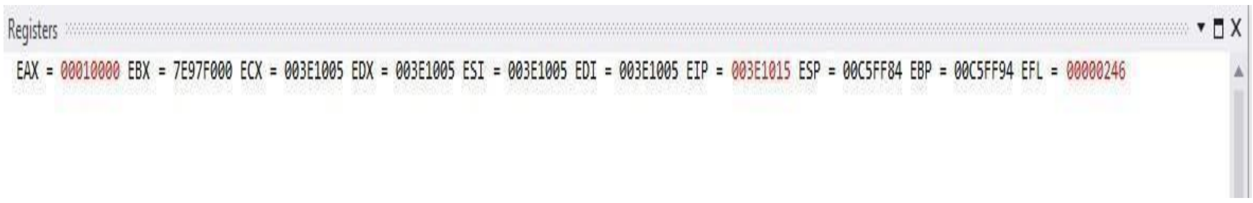


```
EAX = 6801D786 EBX = 7E97F000 ECX = 003E1005 EDX = 003E1005 ESI = 003E1005 EDI = 003E1005 EIP = 003E1010 ESP = 00C5FF84 EBP = 00C5FF94 EFL = 00000244
```

Press **F10** again to execute next line.

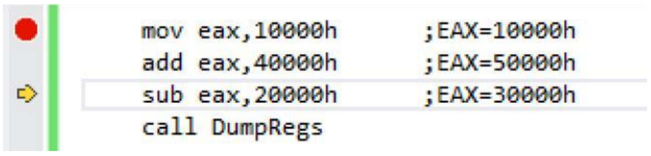


```
mov eax,10000h        ;EAX=10000h
add eax,40000h        ;EAX=50000h
sub eax,20000h        ;EAX=30000h
call DumpRegs
```

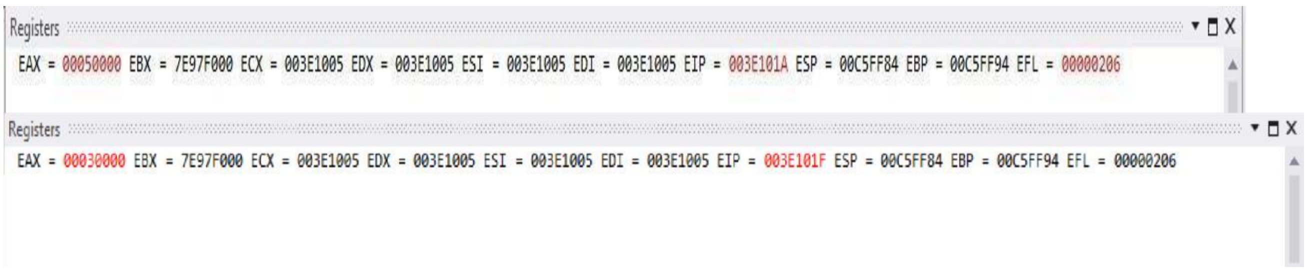


```
EAX = 00010000 EBX = 7E97F000 ECX = 003E1005 EDX = 003E1005 ESI = 003E1005 EDI = 003E1005 EIP = 003E1015 ESP = 00C5FF84 EBP = 00C5FF94 EFL = 00000246
```

Again press **F10** key for next instruction execution.



```
mov eax,10000h        ;EAX=10000h
add eax,40000h        ;EAX=50000h
sub eax,20000h        ;EAX=30000h
call DumpRegs
```



```
Registers
EAX = 00050000 EBX = 7E97F000 ECX = 003E1005 EDX = 003E1005 ESI = 003E1005 EDI = 003E1005 EIP = 003E101A ESP = 00C5FF84 EBP = 00C5FF94 EFL = 00000206
Registers
EAX = 00030000 EBX = 7E97F000 ECX = 003E1005 EDX = 003E1005 ESI = 003E1005 EDI = 003E1005 EIP = 003E101F ESP = 00C5FF84 EBP = 00C5FF94 EFL = 00000206
```

Press **F10** again, the program will not terminate after executing the current instruction and as soon as it reaches the line with a call to **DumpRegs**

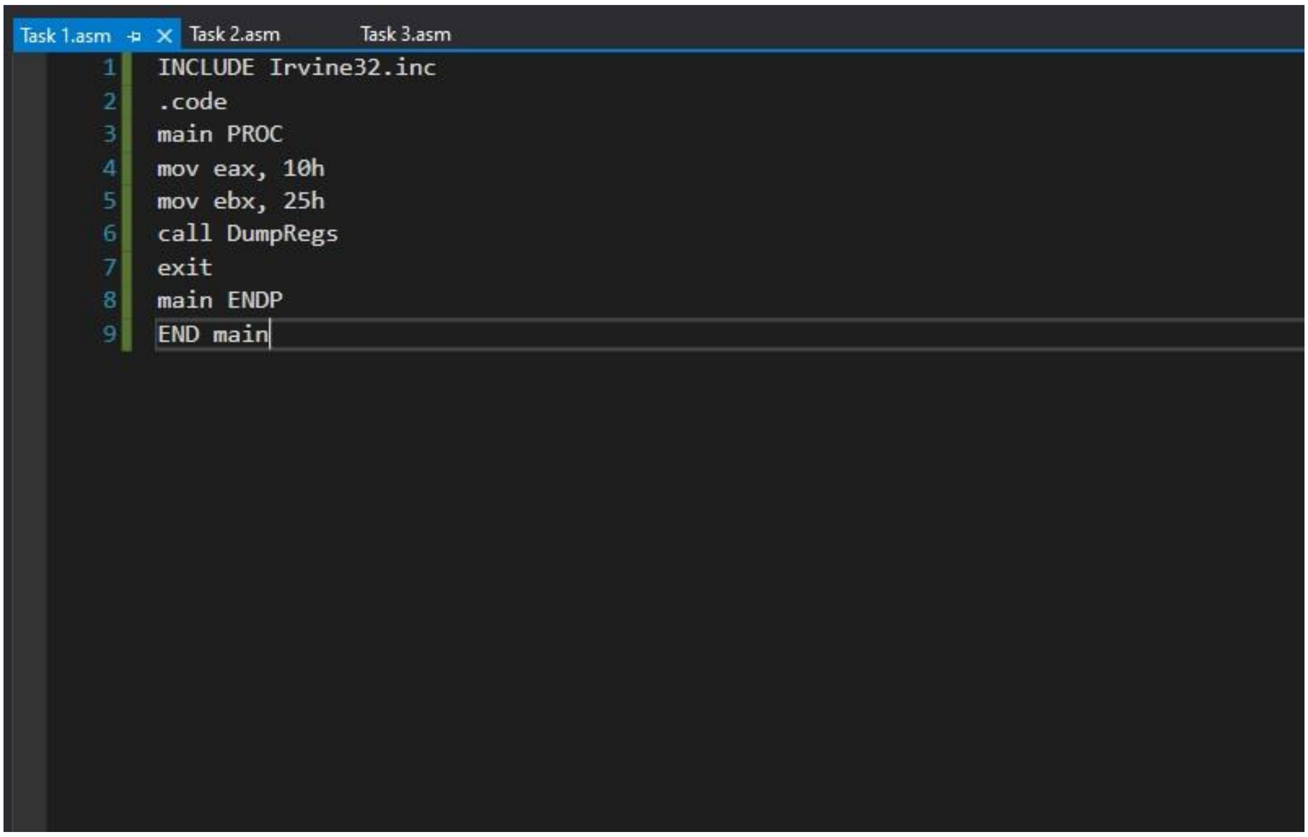
Section 2: Exercise

1. Install Visual Studio 2019 & create a new Visual C++ project for Assembly Language.
2. Configure the project using the steps show in this lab.
3. Run a test program in console window by changing the value of EAX in line 6 to 8500h.
4. Debug the below program and note down the values of all the registers after the execution of each line.

```
TITLE My First Program (Test.asm)
INCLUDE Irvine32.inc

        .code
        main PROC
        mov eax, 47h
        mov ebx, 39h
        mov ecx, 60h
        add eax, ebx add
        eax, ecx mov
        ebx, 85h mov
        ecx, 64h add
        eax, ebx add
        eax, ecx

call DumpRegs
exit
```

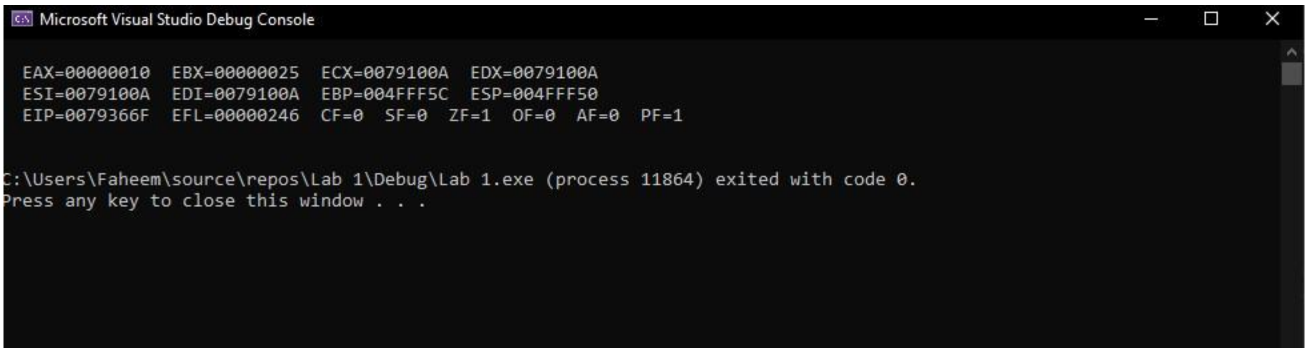
main ENDP END
main

Task 1:



Task 1.asm Task 2.asm Task 3.asm

```
INCLUDE Irvine32.inc
.code
main PROC
mov eax, 10h
mov ebx, 25h
call DumpRegs
exit
main ENDP
END main
```

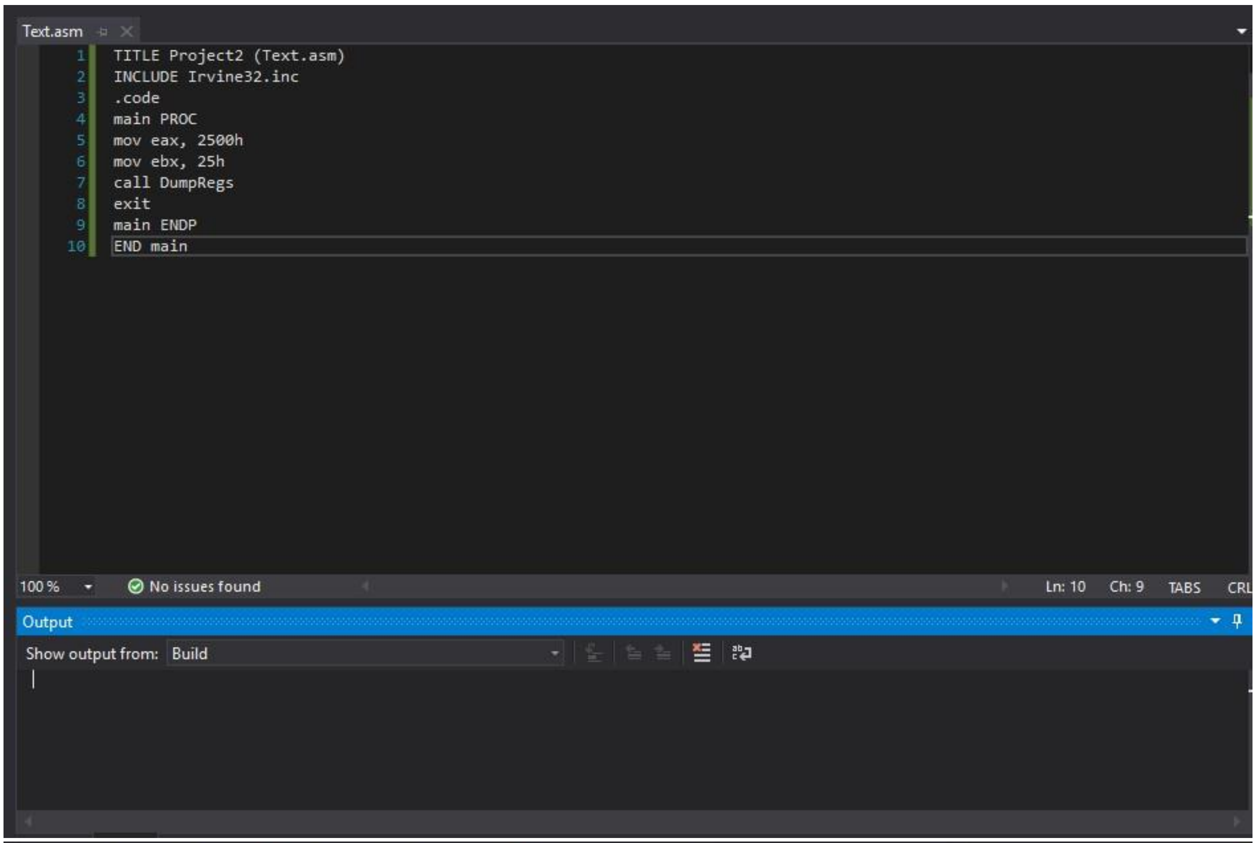


Microsoft Visual Studio Debug Console

```
  EAX=00000010  EBX=00000025  ECX=0079100A  EDX=0079100A
  ESI=0079100A  EDI=0079100A  EBP=004FFF5C  ESP=004FFF50
  EIP=0079366F  EFL=00000246  CF=0  SF=0  ZF=1  OF=0  AF=0  PF=1


C:\Users\Faheem\source\repos\Lab 1\Debug\Lab 1.exe (process 11864) exited with code 0.
Press any key to close this window . . .
```

Task 2:

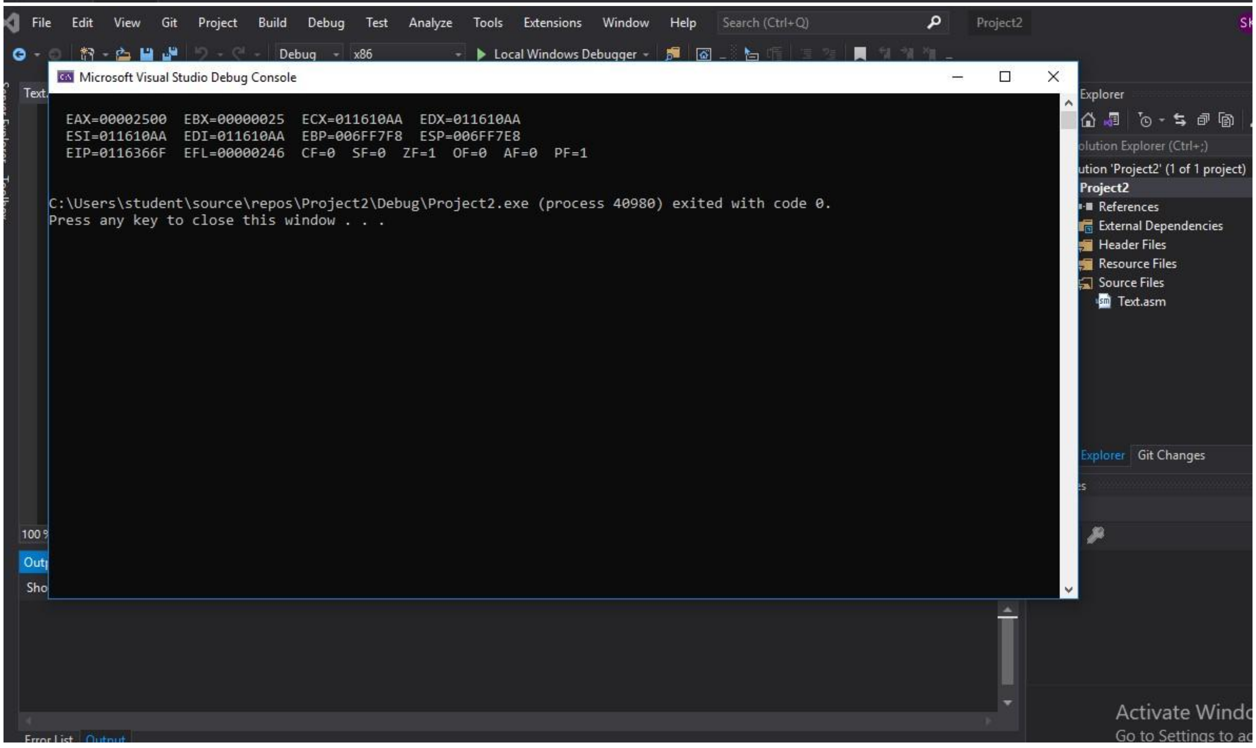
Task 2 includes Run a test program in console window by changing the value of EAX in line 6 to 8500h.



```
TITLE Project2 (Text.asm)
INCLUDE Irvine32.inc
.code
main PROC
mov eax, 2500h
mov ebx, 25h
call DumpRegs
exit
main ENDP
END main
```

No issues found

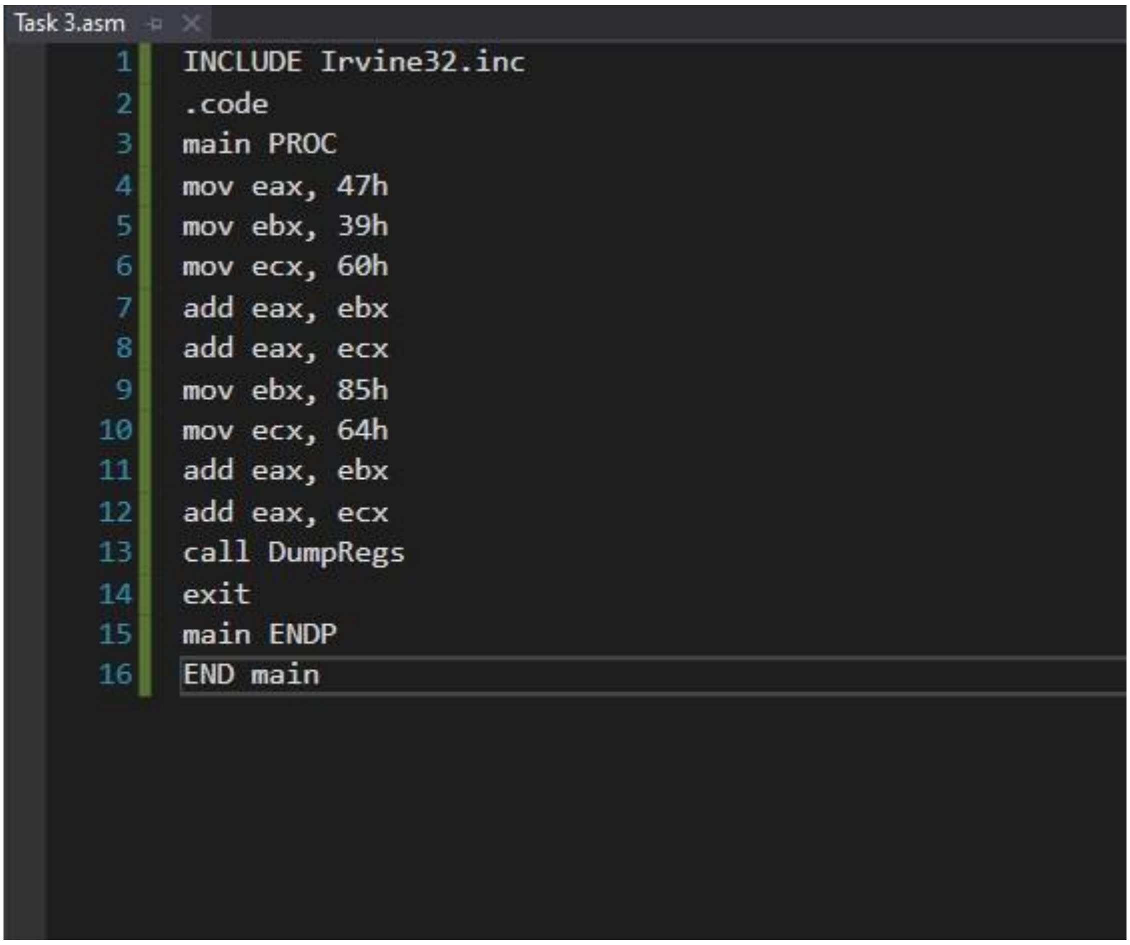
Show output from: Build



Microsoft Visual Studio Debug Console

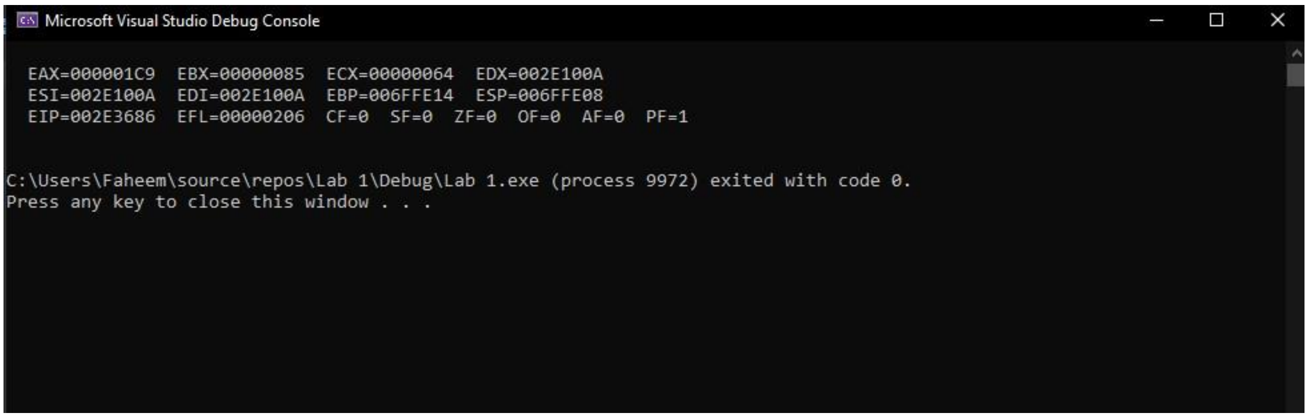
```
  EAX=00002500  EBX=00000025  ECX=011610AA  EDX=011610AA
  ESI=011610AA  EDI=011610AA  EBP=006FF7F8  ESP=006FF7E8
  EIP=0116366F  EFL=00000246  CF=0  SF=0  ZF=1  OF=0  AF=0  PF=1

C:\Users\student\source\repos\Project2\Debug\Project2.exe (process 40980) exited with code 0.
Press any key to close this window . . .
```

Task 3:



Task 3.asm

```
INCLUDE Irvine32.inc
.code
main PROC
mov eax, 47h
mov ebx, 39h
mov ecx, 60h
add eax, ebx
add eax, ecx
mov ebx, 85h
mov ecx, 64h
add eax, ebx
add eax, ecx
call DumpRegs
exit
main ENDP
END main
```

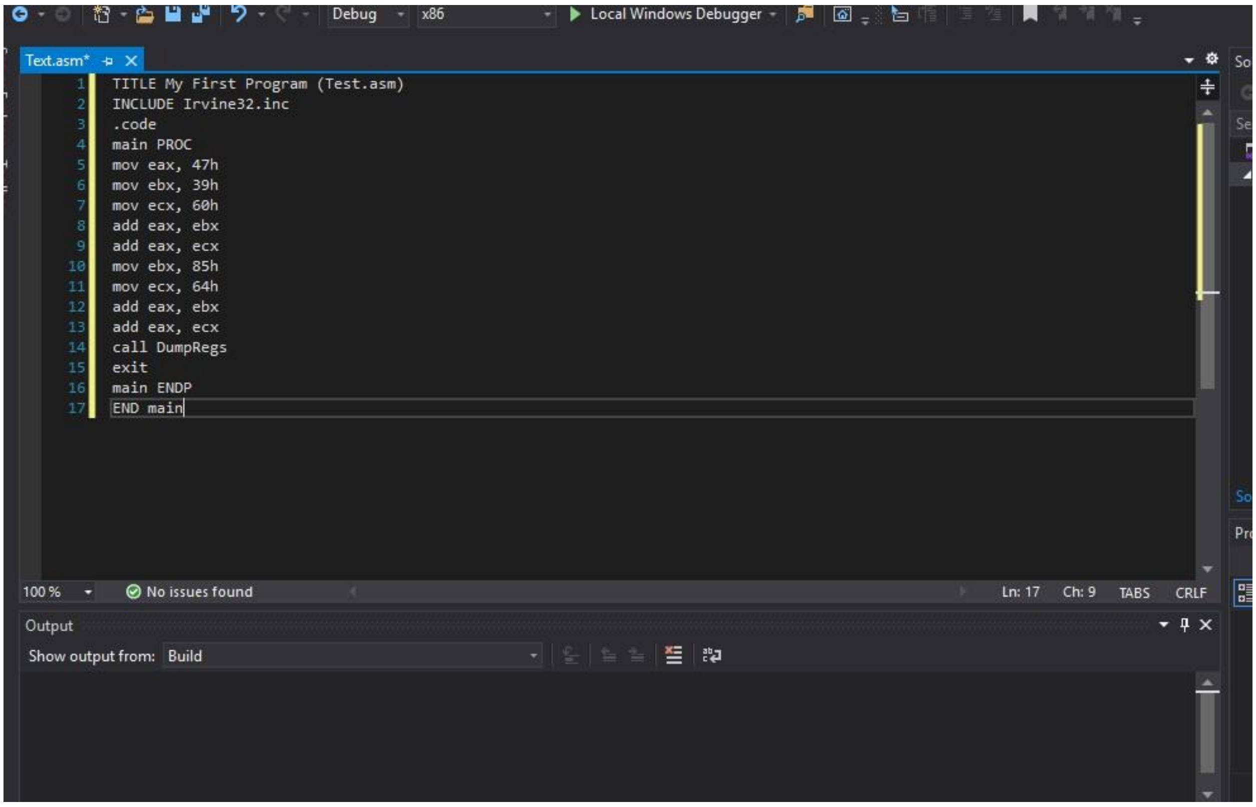


Microsoft Visual Studio Debug Console

```
  EAX=000001C9  EBX=00000085  ECX=00000064  EDX=002E100A
  ESI=002E100A  EDI=002E100A  EBP=006FFE14  ESP=006FFE08
  EIP=002E3686  EFL=00000206  CF=0  SF=0  ZF=0  OF=0  AF=0  PF=1


C:\Users\Faheem\source\repos\Lab 1\Debug\Lab 1.exe (process 9972) exited with code 0.
Press any key to close this window . . .
```

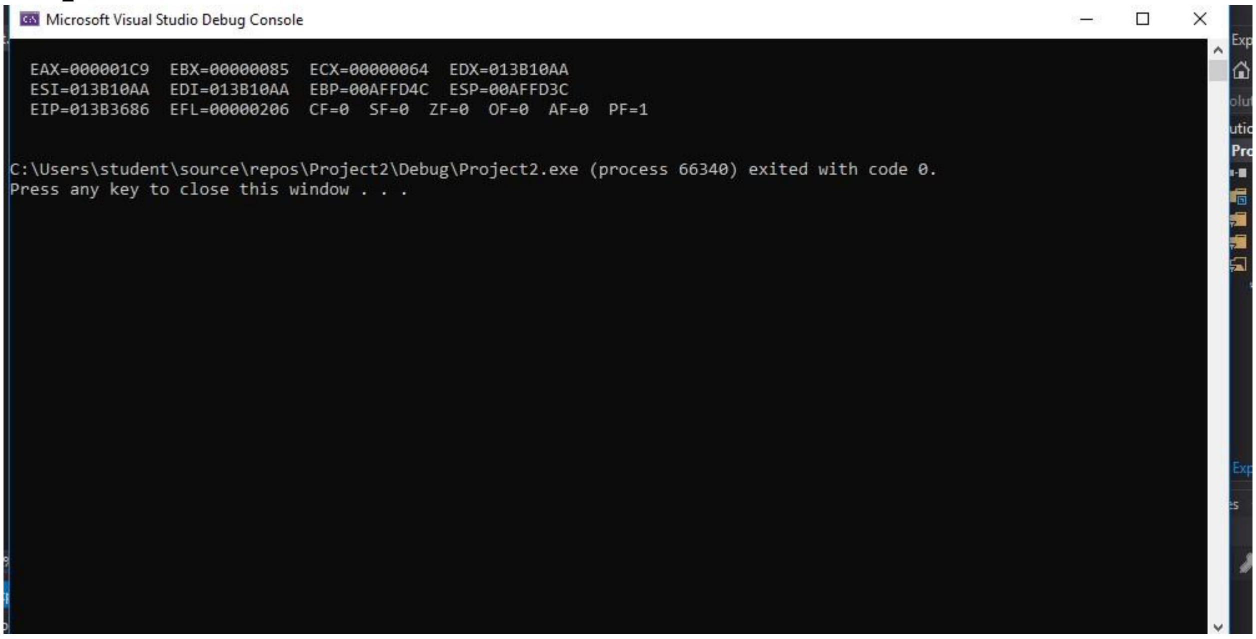
DEBUGGING TASK

Code:



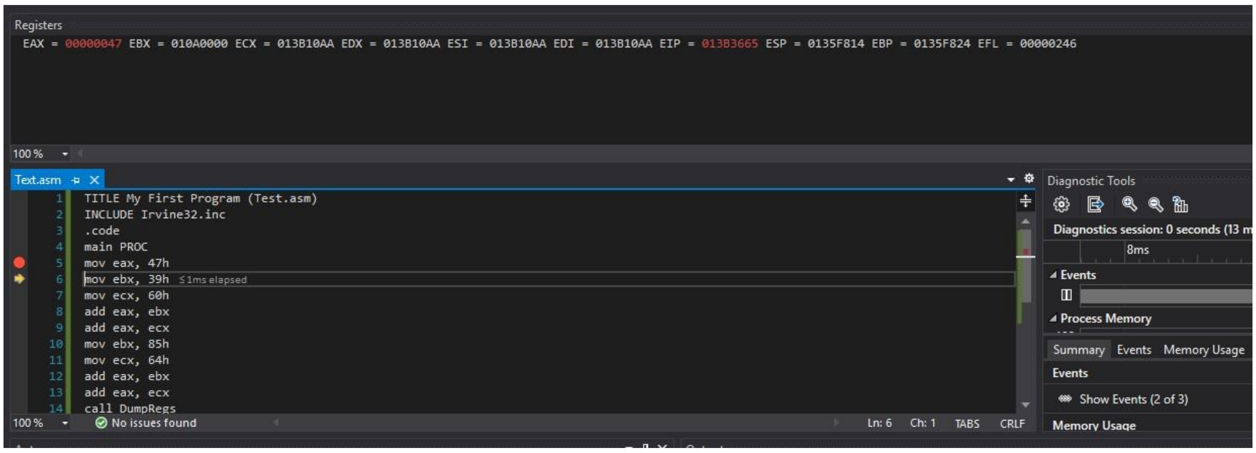
```
TITLE My First Program (Test.asm)
INCLUDE Irvine32.inc
.code
main PROC
mov eax, 47h
mov ebx, 39h
mov ecx, 60h
add eax, ebx
add eax, ecx
mov ebx, 85h
mov ecx, 64h
add eax, ebx
add eax, ecx
call DumpRegs
exit
main ENDP
END main
```

Output:



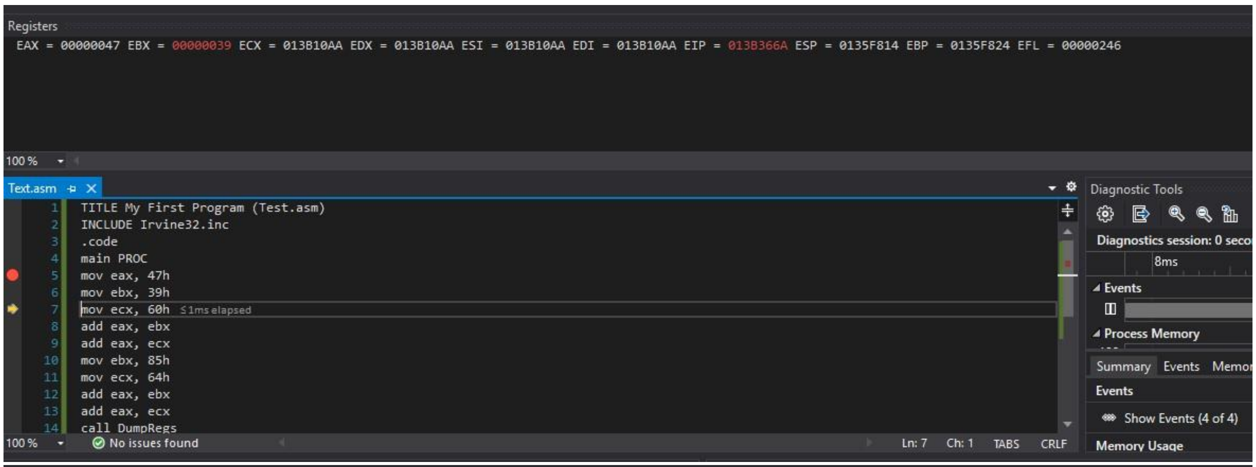
```
  EAX=000001C9  EBX=00000085  ECX=00000064  EDX=013B10AA
  ESI=013B10AA  EDI=013B10AA  EBP=00AFFD4C  ESP=00AFFD3C
  EIP=013B3686  EFL=00000206  CF=0  SF=0  ZF=0  OF=0  AF=0  PF=1


C:\Users\student\source\repos\Project2\Debug\Project2.exe (process 66340) exited with code 0.
Press any key to close this window . . .
```

Debugging now:



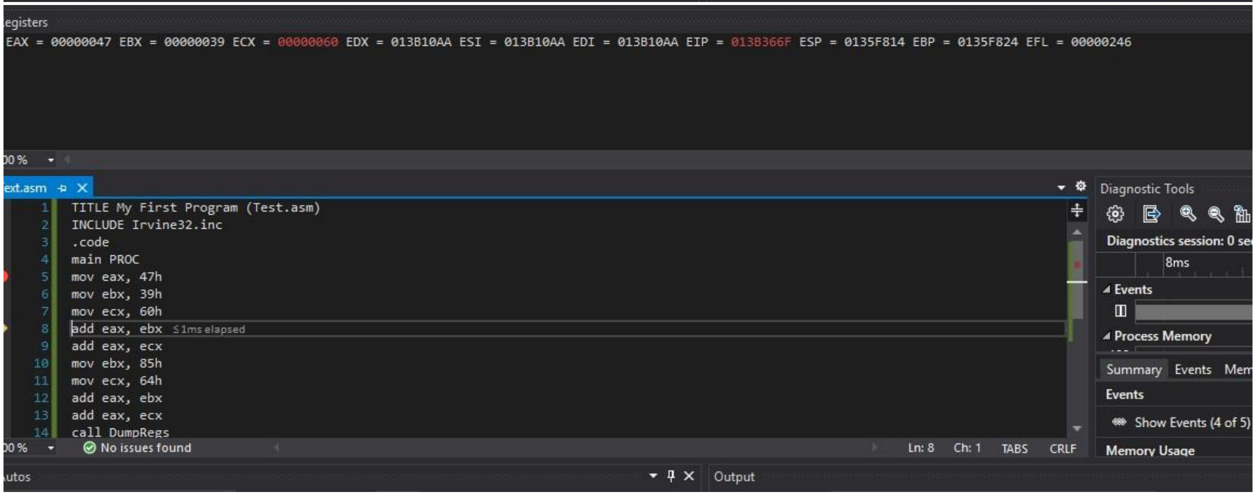
```
Registers
 EAX = 00000047 EBX = 010A0000 ECX = 013B10AA EDX = 013B10AA ESI = 013B10AA EDI = 013B10AA EIP = 013B3665 ESP = 0135F814 EBP = 0135F824 EFL = 00000246
```

```
TITLE My First Program (Test.asm)
INCLUDE Irvine32.inc
.code
main PROC
mov eax, 47h
mov ebx, 39h  ≤1ms elapsed
mov ecx, 60h
add eax, ebx
add eax, ecx
mov ebx, 85h
mov ecx, 64h
add eax, ebx
add eax, ecx
call DumpRegs
```



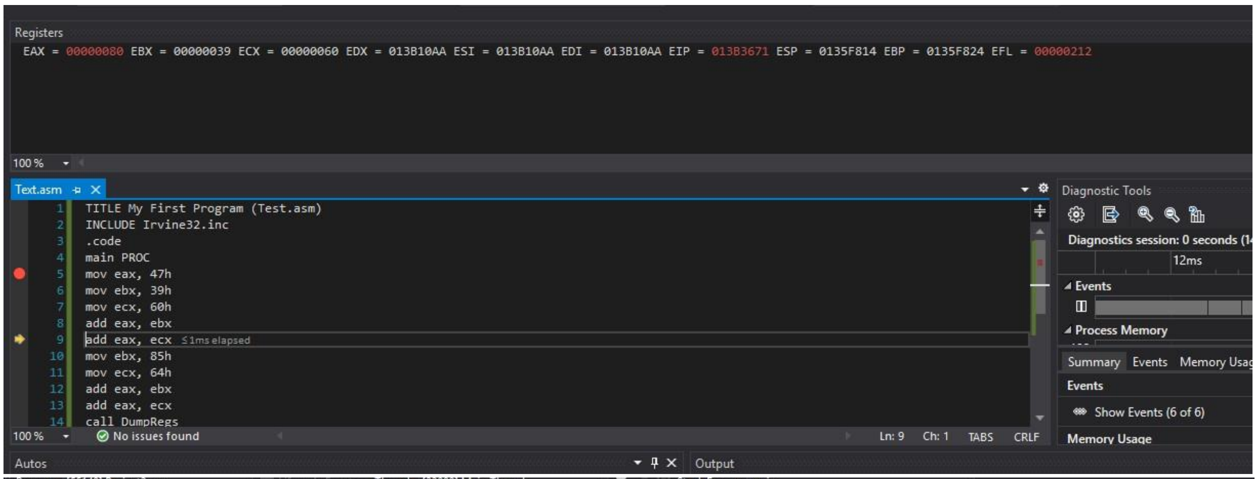
```
Registers
EAX = 00000047 EBX = 00000039 ECX = 013B10AA EDX = 013B10AA ESI = 013B10AA EDI = 013B10AA EIP = 013B366A ESP = 0135F814 EBP = 0135F824 EFL = 00000246
```

```
TITLE My First Program (Test.asm)
INCLUDE Irvine32.inc
.code
main PROC
mov eax, 47h
mov ebx, 39h
mov ecx, 60h  ≤1ms elapsed
add eax, ebx
add eax, ecx
mov ebx, 85h
mov ecx, 64h
add eax, ebx
add eax, ecx
call DumpRegs
```



```
egisters
EAX = 00000047 EBX = 00000039 ECX = 00000060 EDX = 013B10AA ESI = 013B10AA EDI = 013B10AA EIP = 013B366F ESP = 0135F814 EBP = 0135F824 EFL = 00000246
```

```
TITLE My First Program (Test.asm)
INCLUDE Irvine32.inc
.code
main PROC
mov eax, 47h
mov ebx, 39h
mov ecx, 60h
add eax, ebx  ≤1ms elapsed
add eax, ecx
mov ebx, 85h
mov ecx, 64h
add eax, ebx
add eax, ecx
call DumpRegs
```

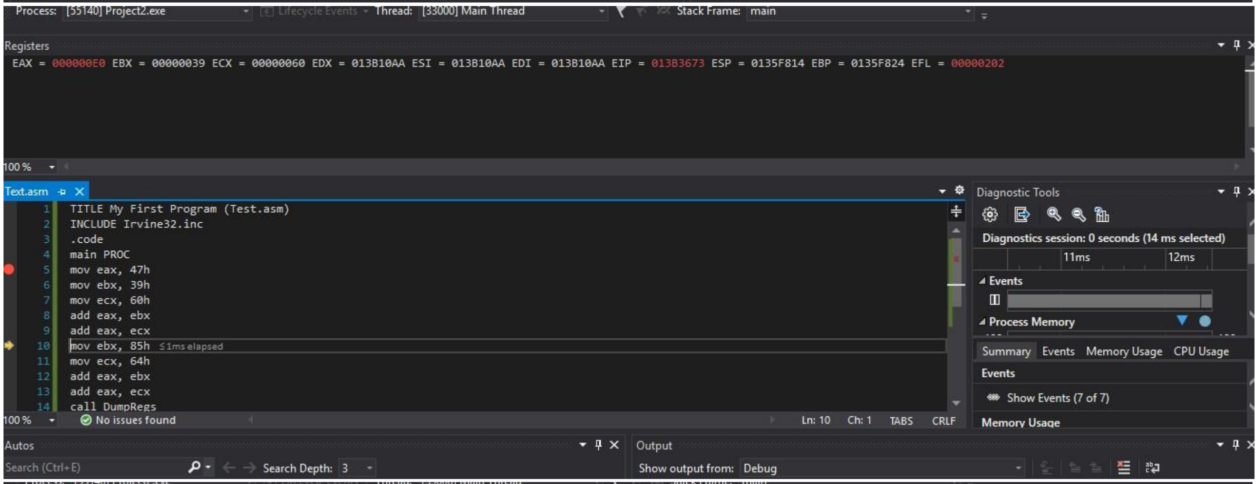


Registers
```
EAX = 00000080 EBX = 00000039 ECX = 00000060 EDX = 013B10AA ESI = 013B10AA EDI = 013B10AA EIP = 013B3671 ESP = 0135F814 EBP = 0135F824 EFL = 00000212
```

Text.asm
```
TITLE My First Program (Test.asm)
INCLUDE Irvine32.inc
.code
main PROC
mov eax, 47h
mov ebx, 39h
mov ecx, 60h
add eax, ebx
add eax, ecx  ≤1ms elapsed
mov ebx, 85h
mov ecx, 64h
add eax, ebx
add eax, ecx
call DumpRegs
```
100 % No issues found Ln: 9 Ch: 1 TABS CRLF

Diagnostic Tools
Diagnostics session: 0 seconds (1
12ms
Events
Process Memory
Summary Events Memory Usag
Events
Show Events (6 of 6)
Memory Usage

Autos Output



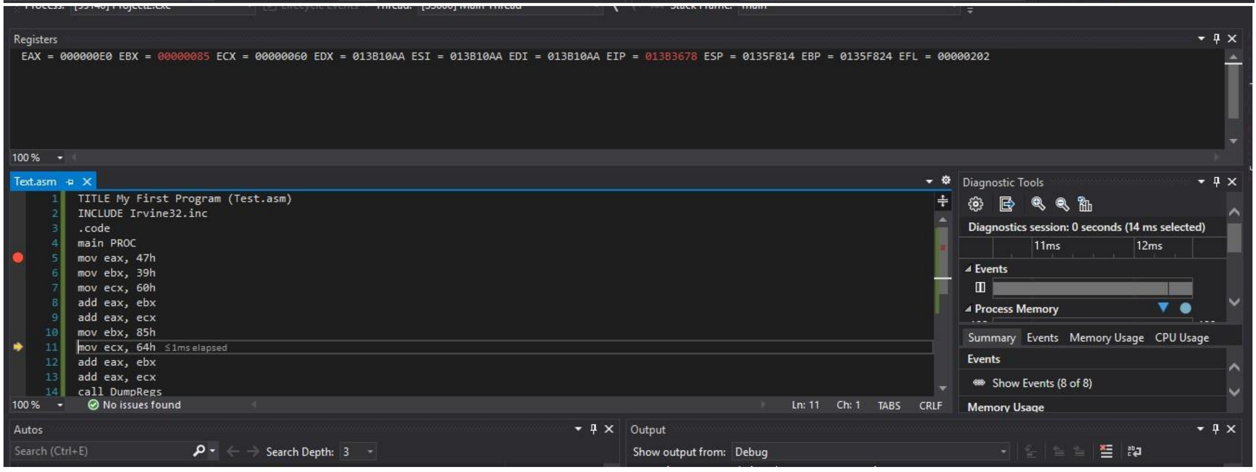
Process: [55140] Project2.exe Lifecycle Events Thread: [33000] Main Thread Stack Frame: main

Registers
```
EAX = 000000E0 EBX = 00000039 ECX = 00000060 EDX = 013B10AA ESI = 013B10AA EDI = 013B10AA EIP = 013B3673 ESP = 0135F814 EBP = 0135F824 EFL = 00000202
```

Text.asm
```
TITLE My First Program (Test.asm)
INCLUDE Irvine32.inc
.code
main PROC
mov eax, 47h
mov ebx, 39h
mov ecx, 60h
add eax, ebx
add eax, ecx
mov ebx, 85h  ≤1ms elapsed
mov ecx, 64h
add eax, ebx
add eax, ecx
call DumpRegs
```
100 % No issues found Ln: 10 Ch: 1 TABS CRLF

Diagnostic Tools
Diagnostics session: 0 seconds (14 ms selected)
11ms 12ms
Events
Process Memory
Summary Events Memory Usage CPU Usage
Events
Show Events (7 of 7)
Memory Usage

Autos
Search (Ctrl+E) Search Depth: 3
Output
Show output from: Debug

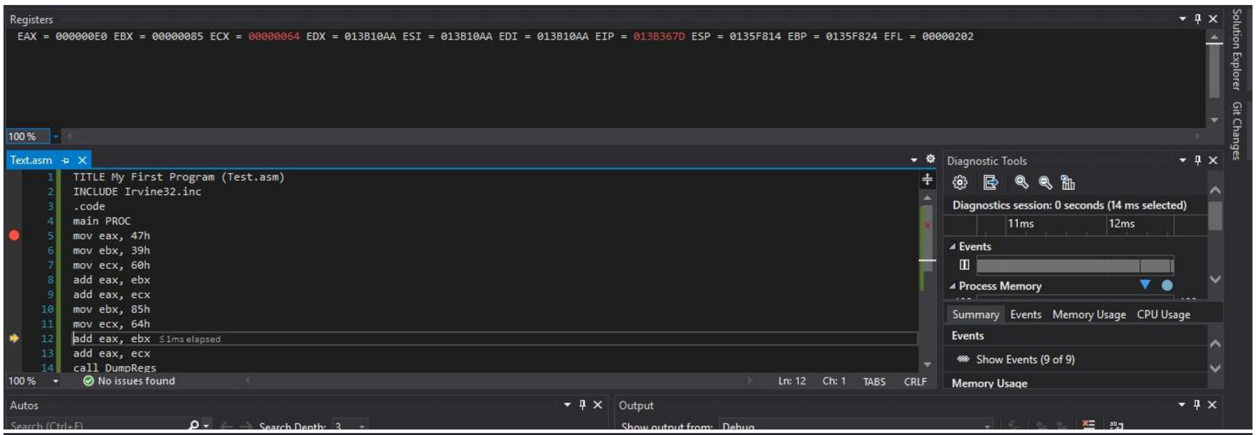


Registers
```
EAX = 000000E0 EBX = 00000085 ECX = 00000060 EDX = 013B10AA ESI = 013B10AA EDI = 013B10AA EIP = 013B3678 ESP = 0135F814 EBP = 0135F824 EFL = 00000202
```

Text.asm
```
TITLE My First Program (Test.asm)
INCLUDE Irvine32.inc
.code
main PROC
mov eax, 47h
mov ebx, 39h
mov ecx, 60h
add eax, ebx
add eax, ecx
mov ebx, 85h
mov ecx, 64h  ≤1ms elapsed
add eax, ebx
add eax, ecx
call DumpRegs
```
100 % No issues found Ln: 11 Ch: 1 TABS CRLF

Diagnostic Tools
Diagnostics session: 0 seconds (14 ms selected)
11ms 12ms
Events
Process Memory
Summary Events Memory Usage CPU Usage
Events
Show Events (8 of 8)
Memory Usage

Autos
Search (Ctrl+E) Search Depth: 3
Output
Show output from: Debug

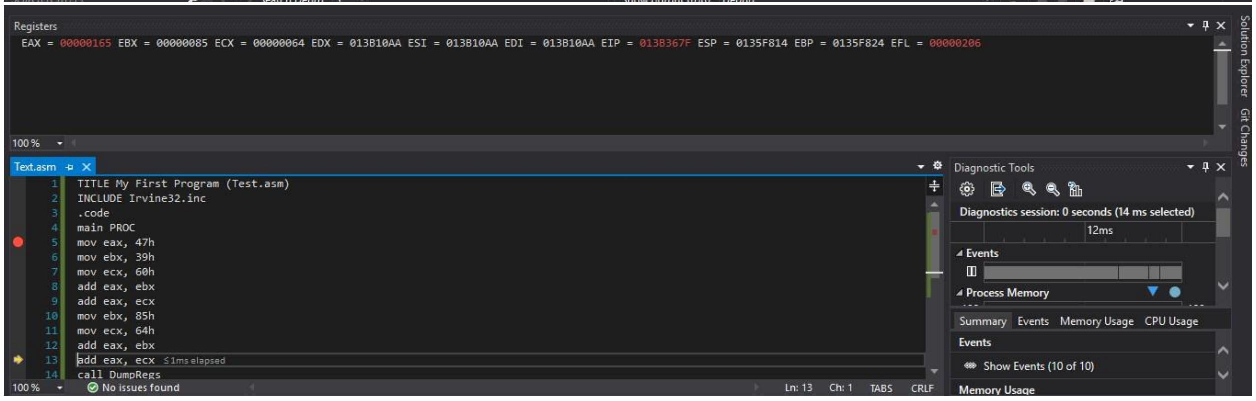


Registers

```
EAX = 000000E0 EBX = 00000085 ECX = 00000064 EDX = 013B10AA ESI = 013B10AA EDI = 013B10AA EIP = 013B367D ESP = 0135F814 EBP = 0135F824 EFL = 00000202
```

Text.asm

```
TITLE My First Program (Test.asm)
INCLUDE Irvine32.inc
.code
main PROC
mov eax, 47h
mov ebx, 39h
mov ecx, 60h
add eax, ebx
add eax, ecx
mov ebx, 85h
mov ecx, 64h
add eax, ebx  ≤1ms elapsed
add eax, ecx
call DumpRegs
```



Registers

```
EAX = 00000165 EBX = 00000085 ECX = 00000064 EDX = 013B10AA ESI = 013B10AA EDI = 013B10AA EIP = 013B367F ESP = 0135F814 EBP = 0135F824 EFL = 00000206
```

Text.asm

```
TITLE My First Program (Test.asm)
INCLUDE Irvine32.inc
.code
main PROC
mov eax, 47h
mov ebx, 39h
mov ecx, 60h
add eax, ebx
add eax, ecx
mov ebx, 85h
mov ecx, 64h
add eax, ebx
add eax, ecx  ≤1ms elapsed
call DumpRegs
```